

# Real-time Algorithm for Self-Reflective Model Predictive Control

Xuhui Feng, Boris Houska

*School of Information Science and Technology, ShanghaiTech University, China.*  
borish@shanghaitech.edu.cn

---

## Abstract

This paper is about a real-time model predictive control (MPC) algorithm for a particular class of model based controllers, whose objective consists of a nominal tracking objective and an additional learning objective. Here, the construction of the learning term is based on economic optimal experiment design criteria. It is added to the MPC objective in order to excite the system from time-to-time on purpose in order to improve the accuracy of the state and parameter estimates in the presence of incomplete or noise affected measurements. A particular focus of this paper is on so-called self-reflective model predictive control schemes, which have the property that the additional learning term can be interpreted as the expected loss of optimality of the controller in the presence of random measurement errors. The main contribution of this paper is a formulation-tailored algorithm, which exploits the particular structure of self-reflective MPC problems in order to speed-up the online computation. It is shown that, in contrast to generic state-of-the-art optimal control problem solvers, the proposed algorithm can solve the self-reflective optimization problems with reasonable additional computational effort and in real-time. The advantages of the proposed real-time scheme are illustrated by applying the algorithm to a nonlinear process control problem in the presence of measurement errors and process noise.

**Keywords:** Optimal Control, Optimal Experiment Design, Model Predictive Control

---

## 1. Introduction

The standard variant of model predictive control (MPC) [28, 37] relies on the principle of *certainty equivalence*: at every sampling time a nominal control performance objective is optimized subject to dynamic model equations as well as

control and state constraints over a finite prediction horizon under the assumption that there are neither state estimation errors, nor external disturbances, nor any kind of model plant mismatches present, although all these errors and disturbances are the reason why a feedback controller is needed in the first place. After the MPC controller sends its first control input to the real process, the next optimization problem is solved by using the latest state estimate in order to close the loop. The success of such certainty equivalent model predictive controllers, also in industrial applications [35], is to a large part due to the availability of fast and reliable real-time optimal control problem solvers [6, 45]. During the last years algorithms as well as mature automatic code generation based software have been developed, which can solve nonlinear model predictive control problems online and within sampling times in the milli- and microsecond range [19, 27].

Although one might argue that standard MPC and its more traditional variants do not attempt to achieve a tradeoff between nominal performance as well as learning objectives, many other controllers, which implement such tradeoffs, have a longer history. In fact, during the last 50 years, there have been many suggestions on how to develop controllers that implement such a tradeoff between control performance and learning. One of the pioneers of the so-called dual control problem is A. Feldbaum, who published a whole series of seminal papers on this topic [8]. Feldbaum’s original dual control problem formulation is based on an optimization problem that minimizes the expected control performance, typically a least-squares tracking term, under the assumption that the accuracy of the parameter estimates depends on the information content of future state measurements. Unfortunately, explicit solutions for this original problem formulation of the dual control problem have so far only been found for very simple problems with one state variable. Numerical algorithms for solving the dual control problem in higher dimensional spaces, e.g., based on approximate dynamic programming, turn out to be rather expensive [24]. For an overview about other attempts to solve the dual control problem approximately by using techniques from the field of adaptive control the reader is referred to the overview articles [10, 42].

As mentioned above, earlier or more traditional variants of MPC usually do not analyze learning objectives explicitly. However, during the last decade this situation has changed and, especially in recent years, there have appeared a significant number of articles about MPC variants that incorporate additional learning terms in order to achieve better future state and parameter estimates. For example, in [18] it is suggested to augment the standard MPC objective by an additional

term that penalizes an approximation of the variance of future state estimates. This can be implemented by augmenting the model equations with an extended Kalman filter that can be used to predict the variance of future state and parameter estimates in a linear approximation. Similar extensions of MPC with learning terms have been proposed in [14, 15], which augment the nominal MPC objective with optimal experiment design objectives that penalize the predicted variance of the system parameters. In recent years there have appeared a number of articles on persistently exciting MPC [16, 25, 31, 39, 44], which all discuss different ways to excite model predictive controllers in order to improve the accuracy of future state and parameter estimates. An even more recent trend is to extend the concept of application oriented optimal experiment design [17] by associated terms in the objective or constraints of an MPC problem [22, 23]. Moreover, a recent paper on self-reflective model predictive control [21] proposes a model predictive controller that minimizes a prediction of its own expected loss of control performance in the presence of measurement noise. Notice that all these developments are closely related to the research on output-feedback MPC [11]. For example, in [29, 30] it is suggested to augment the dynamic system with a differential equation that implements a state estimator and then use methods from the field of robust MPC to robustly control the coupled system of the controller and estimator. This leads on the one hand to a rigorous framework for bounding or approximating the influence of the state estimator in MPC but, on the other hand, using robust tube based MPC methods also adds another layer of complexity to a potentially nonlinear dynamics that is already challenging to solve.

A rather apparent drawback of all the above reviewed model predictive control schemes with additional learning objectives is that they are based on introducing additional, typically matrix-valued hyperstates, which are needed for predicting the accuracy of future state and parameter estimates. This leads to an optimal control problem that is from a numerical computation perspective much more difficult to solve than the corresponding optimal control problem without learning terms. Unfortunately, real-time algorithms, which exploit the structure of the model predictive control problems with additional learning terms, are, as far as the authors are aware, not available to date—let alone implementations and software for solving these problems reliably. Therefore, a principal goal of this paper is to develop a real-time algorithm that can exploit the structure of such problems. Here, we focus on the self-reflective model predictive control formulation, which has been proposed in [21] and which is based on augmenting a nominal MPC objective with an economic optimal experiment design criterion [20]. However,

the methods in this paper can also be extended for most of the other integrated experiment design MPC methods that have been reviewed above. The main contribution of this paper is the development of a novel real-time algorithm for model predictive control with additional learning objectives. In order to avoid confusion, we mention here that the formulation of such augmented MPC problems is not an original contribution of this paper, since there are many articles about how to formulate such problems as reviewed above. The main motivation of this paper is based on the fact that the practical intention of adding a (small) correction term for the purpose of learning is, at least in most applications that are known to the authors, to obtain a refinement of an existing model based control scheme. A major change of the properties of a standard MPC scheme is usually not intended when such learning refinements are added. However, at the current status of research adding such a small correction term for the purpose of learning means—at least based on the authors’ numerical experience—an increase of computation time in the order of a factor of 100 – 1000 compared to standard MPC, if one of the above reviewed augmented MPC formulations is implemented by using a generic optimal control problem solver. From a practical perspective such an increase of run-time is hardly acceptable and might in fact be the main reason why MPC controllers with additional learning objectives, as developed during the last decade, are not yet used more widely in the process control industry. In order to remedy this situation, the current paper develops a tailored algorithm that can solve the self-reflective MPC problem within a sampling time that amounts to less than four times the sampling time of an associated certainty-equivalent real-time MPC scheme [6]. The corresponding implementation is based on an extension of the automatic code generation tools that are implemented by using CASADI [1, 2] and ACADO toolkit [19], which can be used to implement self-reflective MPC schemes with a sampling time of less than a millisecond.

### 1.1. Notation

We use the symbol  $S_+^n$  to denote the set of positive semi-definite  $n \times n$  matrices. Similarly,  $S_{++}^n$  denotes the set of positive definite  $n \times n$  matrices. Throughout this paper, the symbol  $k \in \{0, \dots, N-1\}$  is used as a running index, which, by convention, always runs from 0 to  $N-1 \in \mathbb{N}$ . For example, if we write a discrete-time system in the form

$$x_{k+1} = f(x_k, u_k) ,$$

then this means—if not explicitly stated otherwise—that this equation should hold for all  $k \in \{0, \dots, N-1\}$ .

## 2. Review of real-time algorithms for nonlinear MPC

This section introduces notation and reviews an existing real-time algorithm for standard, certainty equivalent MPC, which is the basis for the developments in this paper.

### 2.1. Certainty equivalent MPC

Standard nonlinear MPC [37] solves at each sampling time an optimization problem of the form

$$\begin{aligned} \min_{x,u} \quad & \sum_{k=0}^{N-1} l(x_k, u_k) + m(x_N) \\ \text{s.t.} \quad & \begin{cases} x_{k+1} = f(x_k, u_k), & x_0 = y \\ \underline{u} \leq u_k \leq \bar{u} \end{cases} \end{aligned} \quad (1)$$

based on the current (potentially inaccurate) state estimate  $y$ , sends the first control input  $u_0$  to the real process, and waits for the next state estimate before the loop is repeated. Here,  $x_k \in \mathbb{R}^{n_x}$  denotes the state,  $u_k \in \mathbb{R}^{n_u}$  the control,  $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x}$  a three-times Lipschitz-continuously differentiable right-hand side function, which satisfies  $f(0,0) = 0$ , and  $\underline{u}, \bar{u} \in \mathbb{R}^{n_u}$  control bounds with  $\underline{u} < 0 < \bar{u}$ , i.e., such that Slater's constraint qualification is satisfied. For simplicity of presentation, we assume that the stage and terminal cost

$$l(x, u) = \frac{1}{2} (x^\top Q x + u^\top R u) \quad \text{and} \quad m(x) = \frac{1}{2} x^\top P_N x$$

are convex quadratic tracking terms with given positive semi-definite weighting matrices  $Q, P_N \in S_+^{n_x}$  and  $R \in S_+^{n_u}$ . However, the considerations in this paper can be extended easily for the case that  $l$  and  $m$  are more general economic objective terms, as long as all functions are three-times Lipschitz-continuously differentiable.

Notice that practical system are often modeled by continuous-time differential equations. However, there exist mature tools from the field of direct optimal control, which can be used to discretize continuous-time optimal control problems leading to a finite dimensional nonlinear programming problem of the above form [3, 4]. More general MPC problem formulations additionally comprise state constraints as well as terminal regions [37], which are however not analyzed in this paper. The reason for this decision is that state constraints can—at least without further precaution—lead to infeasibility, if uncertainties are present. As the goal

of this paper is on the one hand to analyze the influence of random measurement errors, but, on the other hand, is not about robust MPC, state constraints are not considered. Notice that a robustification with respect to state constraints can be taken into account by applying the output-feedback MPC based framework [30], which leads however to a much more complex problem formulation, which can be expected to lead to significantly larger computation times than the methods presented in the current paper.

## 2.2. Real-time algorithm

Real-time Gauss-Newton type algorithms for nonlinear MPC [6] are divided into a *preparation phase* and a *feedback phase*. During the preparation phase, the function  $f$  and its derivatives

$$A(x_k, u_k) = \frac{\partial f(x_k, u_k)}{\partial x} \quad \text{and} \quad B(x_k, u_k) = \frac{\partial f(x_k, u_k)}{\partial u}$$

are evaluated at the predicted state trajectory  $x$  and control input  $u$ . Modern real-time MPC implementations are using advanced algorithmic differentiation methods for evaluating these derivatives efficiently and with high precision [12, 19]. Once the state measurement  $y$  becomes available, the feedback phase starts. During this feedback phase, the quadratic programming problem

$$\begin{aligned} \min_{\Delta x, \Delta u} \quad & \sum_{k=0}^{N-1} l(x_k + \Delta x_k, u_k + \Delta u_k) + m(x_N + \Delta x_N) \\ \text{s.t.} \quad & \begin{cases} \Delta x_0 = y - x_0 \\ \Delta x_{k+1} = A(x_k, u_k)\Delta x_k + B(x_k, u_k)\Delta u_k + f(x_k, u_k) - x_{k+1} \\ \underline{u} \leq u_k + \Delta u_k \leq \bar{u} \end{cases} \end{aligned} \quad (2)$$

is solved recalling that  $l$  and  $m$  are assumed to be convex quadratic forms. Next, the updated control

$$u_{\text{process}} = u_0 + \Delta u_0$$

is sent out to the process. The remaining states and controls are shifted in time and updated as follows:

$$\begin{aligned} \forall k \in \{0, \dots, N-1\}, \quad x_k &\leftarrow x_{k+1} + \Delta x_{k+1}, & x_N &\leftarrow x_N + \Delta x_N, \\ \forall k \in \{0, \dots, N-2\}, \quad u_k &\leftarrow u_{k+1} + \Delta u_{k+1}, & \text{and } u_{N-1} &\leftarrow u_{N-1} + \Delta u_{N-1}. \end{aligned}$$

Notice that there are many variants of the above scheme. For example, in [6] it is suggested to prepare the linear algebra operations that are needed for solving

the QP by applying a condensing based approach that eliminates the optimization variable  $\Delta x$  explicitly. This leads to a smaller but more dense QP to be solved in the feedback phase. A theoretical analysis as well as stability proofs of the above real-time algorithm for nonlinear MPC under mild additional assumptions can be found in [6].

### 2.3. Interior-point versus active set methods

The above real-time algorithm is often applied in combination with active set methods, which can be used for solving the QP (2) efficiently. For example, in [9, 19] online variants of active set methods for linear and nonlinear MPC are proposed, which use tailored hot-start methods in order to further speed-up the computation time during the feedback step. However, other nonlinear MPC implementations, such as [45], use an interior point method based framework. Here, the main idea is to introduce the modified state cost

$$l_\tau(x, u) = l(x, u) - \tau \log[(u - \underline{u})(\bar{u} - u)] ,$$

where  $\tau > 0$  is a barrier parameter. Now, problem (1) can be approximated by an equality-constrained NLP of the form

$$\begin{aligned} \min_{x, u} \quad & \sum_{k=0}^{N-1} l_\tau(x_k, u_k) + m(x_N) \\ \text{s.t.} \quad & \{ x_{k+1} = f(x_k, u_k), \quad x_0 = y, \end{aligned} \tag{3}$$

which is equivalent to problem (1) for the limit  $\tau \rightarrow 0^+$  (see [5] for a proof). The numerical advantage of this NLP is that it does not contain inequalities. A corresponding real-time scheme proceeds analogous to Section 2.2, but the difference is that the stage cost  $l_\tau$  is at every sampling time replaced by a second Taylor order expansion at the previous iterate in order to obtain a real-time Newton-type method that solves equality-constrained QPs in the feedback phase. As the band-structure of this QP can be exploited easily, this leads to an algorithm with complexity  $\mathbf{O}(N)$ , which is fortunate if  $N$  is large [45]. However, a drawback of the interior-point method based real-time scheme compared to active set methods is that hot-starts are more difficult to implement. This is due to the fact that the parameter  $\tau$  has to be adjusted during the iterations. Details about how to implement an associated line-search and how to adjust the tuning parameter  $\tau$  during the iterations can be found in [5, 43, 45]. Finally, notice that both the active-set as well as the interior-point method based real-time algorithm have in common that they do not solve problem (1) to optimality, but merely apply one Gauss-Newton or interior point iteration per sampling time.

### 3. MPC with additional learning objectives

In practical applications the state of a system can often not be measured directly and has to be estimated from measurements. Throughout this paper, the measurements are assumed to satisfy the equation

$$\eta_k = Cx_k + v_k, \quad (4)$$

where  $C \in \mathbb{R}^{n_\eta} \times \mathbb{R}^{n_x}$  is a given matrix,  $v_k \in \mathbb{R}^{n_\eta}$  the random measurement error, and  $\eta_k \in \mathbb{R}^{n_\eta}$  the actual measurement. In a more general situation, the dependence of  $\eta_k$  on  $x_k$  could also be nonlinear or additionally depend on  $u_k$ , but the following considerations can be extended easily for this more general case. An additional complication is that the dynamic system itself may be affected by process noise,

$$x_{k+1} = f(x_k, u_k) + w_k, \quad (5)$$

i.e., the variables  $w_k \in \mathbb{R}^{n_x}$  are random variables that cannot be predicted. In the following we assume that the first and second order moments of  $w_k$  and  $v_k$ , denoted by  $W \in \mathcal{S}_{++}^{n_x}$  and  $V \in \mathcal{S}_{++}^{n_\eta}$ , are given,

$$\mathbb{E}\{w_k\} = 0, \quad \mathbb{E}\{v_k\} = 0, \quad \mathbb{E}\{w_k w_k^\top\} = W, \quad \mathbb{E}\{v_k v_k^\top\} = V. \quad (6)$$

Moreover,  $v_k$  and  $w_k$  are assumed to have bounded support such that  $\|v_k\| \leq \gamma$  as well as  $\|w_k\| \leq \gamma$  for a given radius  $\gamma > 0$ . In the introduction a large number of extensions of MPC has been reviewed, which all attempt to include additional terms in the objective or constraints of the MPC controller in order take the accuracy of future state estimates into account when optimizing the control input. The construction of these additional terms uses ideas from the field of optimal experiment design as reviewed next.

#### 3.1. Optimal experiment design based MPC objectives

Optimal experiment design is a rather mature technology and we refer to the text book [34] for a general overview. As this paper is about dynamic system, it is focussed on an optimal experiment design problem formulation that has been analyzed in [41]. Here, the main idea is to introduce an extended Kalman filter in order to predict the variance matrix of future state estimates, which is based on the discrete-time Riccati recursion<sup>1</sup>

$$\begin{aligned} \Sigma_{k+1} &= F(x_k, u_k, \Sigma_k) \\ &= A(x_k, u_k) \left[ \Sigma_k - \Sigma_k C^\top (C \Sigma_k C^\top + V)^{-1} C \Sigma_k \right] A(x_k, u_k)^\top + W. \end{aligned} \quad (7)$$

---

<sup>1</sup>The matrix  $C \Sigma_k C^\top + V$  is invertible, as we assume that  $V$  is positive definite.



This recursion is started with the variance  $\Sigma_0 = \hat{\Sigma} = \mathbb{E}\{(y - x_0^*)(y - x_0^*)^\top\}$  of the current state estimate  $y$ , where  $x_0^*$  denotes the true (but unknown) initial state. Notice that in our context, the derivative function  $A$  is evaluated at the predicted states and control  $(x_k, u_k)$ , as we use the extended Kalman filter to predict the variances  $\Sigma_k$  of future state estimates without knowing the future measurements  $\eta_k$  yet. This is in contrast to the traditional way of applying the extended Kalman filter in the context of estimation, where the measurements  $\eta_k$  are already available and can thus be used to first update the state estimate and then evaluate  $A$  at the state estimate rather than at the predicted state.

In the next step, a scalar experiment design criterion  $\Psi : S_+^{n_x} \rightarrow \mathbb{R}$  can be used to penalize large variances  $\Sigma_k$  in the MPC objective. In this paper we focus on a particular design criterion based on the weighted A-criterion,  $\Psi_k(\Sigma_k) = \text{Tr}(\alpha_k \Sigma_k)$ ,  $\alpha_0, \dots, \alpha_N \in S_+^{n_x}$ , but in principle also other criteria such as a weighted determinant or maximum eigenvalue (D- and E-criterion) can be used as scalar measures of the size of  $\Sigma_k$ . The following MPC problem formulation with additional learning objective has (in a very similar form) for the first time been proposed in [18]:

$$\begin{aligned} \min_{x, u, \Sigma} \quad & \sum_{k=0}^{N-1} \{l_\tau(x_k, u_k) + \text{Tr}(\alpha_k \Sigma_k)\} + m(x_N) + \text{Tr}(\alpha_N \Sigma_N) \\ \text{s.t.} \quad & \begin{cases} x_{k+1} = f(x_k, u_k), & x_0 = y \\ \Sigma_{k+1} = F(x_k, u_k, \Sigma_k), & \Sigma_0 = \hat{\Sigma}. \end{cases} \end{aligned} \quad (8)$$

In the above MPC formulation, the matrix-valued weights  $\alpha_0, \dots, \alpha_N \in S_+^{n_x}$  can be used to tradeoff between the optimal experiment design objective and the nominal control objective: roughly, small values for the  $\alpha_k$ s lead to better nominal tracking performance while larger values typically lead to more excitation for the purpose of learning. However, tuning these weighting matrices by hand may be cumbersome and is, to a certain extent, ambiguous. An alternative is to automatically compute more natural tradeoff weights by using the self-reflective model predictive control scheme [21] that is reviewed next.

### 3.2. Self-reflective model predictive control

As measurement errors and process noise cannot be predicted, they cause an inevitable loss of control performance when compared to an utopia feedforward controller, which can predict all future measurement errors and process noise. Self-reflective MPC [21] is a controller that minimizes the sum of its nominal performance and a second order approximation of its own expected future loss of optimality compared to an utopia feedback controller that can predict everything.

In order to outline the corresponding online optimization problem formulation, the functions

$$\begin{aligned}
X(x, u, \Omega) &= B(x, u)^\top P A(x, u) + \nabla_{ux}^2 [l_\tau(x, u) + \lambda^\top f(x, u)] \\
Y(x, u, \Omega) &= B(x, u)^\top P B(x, u) + \nabla_{uu}^2 [l_\tau(x, u) + \lambda^\top f(x, u)] \\
Z(x, u, \Omega) &= A(x, u)^\top P A(x, u) + \nabla_{xx}^2 [l_\tau(x, u) + \lambda^\top f(x, u)] \\
\Phi(x, u, \Omega) &= X(x, u, \Omega)^\top Y(x, u, \Omega)^{-1} X(x, u, \Omega)
\end{aligned} \tag{9}$$

as well as

$$G(x, u, \Omega) = \begin{pmatrix} \lambda^\top A(x, u) + x^\top Q \\ Z(x, u, \Omega) - \Phi(x, u, \Omega) \end{pmatrix}^\top \quad \text{and} \quad M(x) = \begin{pmatrix} x^\top P_N \\ P_N \end{pmatrix}^\top$$

are introduced. Here, the functions  $X, Y, Z, G$ , and  $M$  are defined for all  $x \in \mathbb{R}^{n_x}$  all  $u \in \mathbb{R}^{n_u}$ , and all  $\Omega = [\lambda, P] \in \mathbb{R} \times S_+^{n_x}$ . Notice that the evaluation of the functions  $X, Y$ , and  $Z$  requires the computation of second order derivatives of the Hamiltonian function  $\mathcal{H} = l_\tau + \lambda^\top f$ . Although there exist efficient second order algorithmic differentiation schemes [36], the computation of these derivatives is expensive. Also notice that the function  $l_\tau$  is strictly convex for any  $\tau > 0$ . Thus, it follows from the second order optimality conditions for problem (1), in this case a discrete-time variant of Pontryagin's maximum principle [33], that the second order derivative of the Hamiltonian  $\nabla_{uu}^2 \mathcal{H}$  is a positive definite function in the neighborhood of a solution of (1). Consequently,  $Y(x, u, \Omega)$  is a positive definite (and thus invertible) matrix function in this neighborhood. Now, self-reflective MPC solves at each sampling time an optimization problem of the form

$$\begin{aligned}
\min_{x, u, \Sigma, \Omega} \quad & \sum_{k=0}^{N-1} \left\{ l_\tau(x_k, u_k) + \frac{1}{2} \text{Tr}(\Phi(x_k, u_k, \Omega_{k+1}) \Sigma_k) \right\} + m(x_N) \\
\text{s.t.} \quad & \begin{cases} x_{k+1} = f(x_k, u_k), & x_0 = y \\ \Sigma_{k+1} = F(x_k, u_k, \Sigma_k), & \Sigma_0 = \hat{\Sigma} \\ \Omega_k = G(x_k, u_k, \Omega_{k+1}), & \Omega_N = M(x_N) \end{cases}
\end{aligned} \tag{10}$$

At this point, one might argue that the optimization problem (10) is very similar to problem (8) in the sense that if we set  $\alpha_k = \frac{1}{2} \Phi(x_k, u_k, \Omega_{k+1})$  and  $\alpha_N = 0$  in problem (8) the objective functions of the two problems coincide. However, notice that in the self-reflective problem (10) the weights  $\alpha_k = \frac{1}{2} \Phi(x_k, u_k, \Omega_{k+1})$  are non-trivial functions of  $x_k$  and  $u_k$  and require the computation of the ancillary backward recursion for the variables  $\Omega_k$ , i.e., the tradeoff between the nominal tracking performance and the additive learning term is optimized. Notice

that the term  $\frac{1}{2}\text{Tr}(\Phi(x_k, u_k, \Omega_{k+1})\Sigma_k)$  approximates the self-reflective MPC controller's own inherent expected loss of optimality in the presence of measurement errors and process noise up to terms of order  $\mathbf{O}(\gamma^3)$ , as established in [21]; see also [40] for a discussion and interpretation of this term in the context of unconstrained linear stochastic control. Other advantages of the self-reflective MPC formulation as well as general economic optimal experiment design criteria are discussed in [20, 21].

### 3.3. Numerical challenges

Unfortunately, solving the integrated experiment design based MPC problem (8) is much more expensive than solving the nominal MPC formulation (3). One reason for this increase in complexity is that problem (8) comprises

$$n_x + \frac{1}{2}n_x(n_x + 1)$$

state variables assuming that the symmetry of the matrix-valued state  $\Sigma_k$  is exploited. The complexity of solving the self-reflective optimization problem (10) is even worse, as this problem comprises

$$2n_x + n_x(n_x + 1)$$

states assuming again that symmetry is already exploited. Another reason is that the forward propagation of the variance matrices  $\Sigma_k$  involves the evaluation of the derivative functions  $A$  and  $B$  of  $f$ , which is often much more expensive than a nominal evaluation of  $f$ . The backward propagation of the state  $\Omega_k$  in the self-reflective MPC problem (10) requires second derivatives of  $f$ , which is expensive, too. Moreover, the stage costs of both (8) as well as (10) are non-convex while the stage cost of (3) consists of a convex tracking term and, optionally, an additive self-concordant and strictly convex barrier function. As if these numerical issues would not be enough, problem (10) comes along with the additional complication that the term  $\frac{1}{2}\text{Tr}(\Phi(x_k, u_k, \Omega_{k+1})\Sigma_k)$  couples the forward states with index  $k$ , namely  $x_k$  and  $\Sigma_k$ , with the backward state, namely  $\Omega_{k+1}$ , with index  $k + 1$ . This destroys the separability of the objective function and renders existing real-time MPC algorithms not directly applicable, as these existing methods rely on the separability of the objective function. From a process control engineering perspective one might argue that the main motivation for adding a learning term in the MPC objective is (if this is intended at all) that small corrections, usually in the form of minor excitations from the nominal path, are induced in order to

improve future state estimates. However, if adding this minor correction comes along with major numerical difficulties and a huge increase in terms of computation time, its usefulness must be assessed critically. Therefore, the goal of this paper is to develop a tailored real-time algorithm based on the self-reflective MPC problem (10), which addresses the above mentioned numerical challenges in such a way that only a moderate increase in computation time is encountered when adding the learning term to the MPC objective. The hope is that this will help to promote a wider use of integrated experiment design based MPC formulations in practical and industrial process control applications.

#### 4. Real-time algorithm for self-reflective MPC

This section develops a real-time algorithm for self-reflective MPC based on the optimization problem (10).

##### 4.1. Motivation

The following parametric optimization problem can be interpreted as a linearly perturbed version of the original NLP (3):

$$\begin{aligned} \min_{x,u} \quad & \sum_{k=0}^{N-1} \{l_{\tau}(x_k, u_k) + \sigma_k^T u_k\} + m(x_N) \\ \text{s.t.} \quad & \{ x_{k+1} = f(x_k, u_k), \quad x_0 = y. \end{aligned} \quad (11)$$

Here, the parameter vector  $\sigma \in \mathbb{R}^{(N-1)n_u}$  can be used to scale the perturbation. Clearly, for a given  $\sigma$  the cost for solving optimization problem (11) can be expected to be basically the same as the cost for solving the NLP (3), as an additional linear term in the objective hardly adds numerical difficulties. In order to avoid confusion at this point, it is mentioned that the method presented here should not be mixed up with the so-called *modifier adaption method* [26], although one might argue that the parameter  $\sigma$  could be interpreted as a “modifier” that corrects the stage cost. However, in the context of the modifier adaption method, the correction of the objective is added in order to correct model-plant mismatches and  $\sigma$  is updated based on measurements of the objective gradient and constraints violation. In contrast to this approach, the intention of the presented framework is different, as we intend to perturb the gradient of the stage cost in order to improve the future state estimates while unstructured model-plant mismatches are beyond the scope of the current paper. In order to establish a connection between

problem (11) and the self-reflective MPC controller (10) the following auxiliary function is introduced:

$$E(u, x_0, \Sigma_0) = \min_{x, \Sigma, \Omega} \sum_{k=0}^{N-1} \frac{1}{2} \text{Tr}(\Phi(x_k, u_k, \Omega_{k+1}) \Sigma_k) \quad (12)$$

$$\text{s.t.} \quad \begin{cases} x_{k+1} = f(x_k, u_k), \\ \Sigma_{k+1} = F(x_k, u_k, \Sigma_k), \\ \Omega_k = G(x_k, u_k, \Omega_{k+1}), \quad \Omega_N = M(x_N). \end{cases}$$

Next, the equivalence of the optimization problems (10) and (11) can be established under the following conditions.

**Lemma 1.** *Let  $f$  be a three-times Lipschitz-continuously differentiable function,  $y = \mathbf{O}(\gamma)$  and  $\|\hat{\Sigma}\| = \mathbf{O}(\gamma^2)$ , and let  $(x^*, u^*, \Sigma^*, \Omega^*)$  be a regular local minimizer of the self-reflective optimization problem (10). If  $\gamma$  is sufficiently small, then the function  $E$  is differentiable in a neighborhood of  $u^*$  and the gradient of  $E$  with respect to  $u$  is Lipschitz continuous. Moreover, if we set  $\sigma = \nabla_u E(u^*, y, \hat{\Sigma})$ , then  $(x^*, u^*)$  is a regular local minimizer of the optimization problem (11).*

*Proof.* Let us introduce the auxiliary function

$$L(u, x_0) = \min_x \sum_{k=0}^{N-1} l_\tau(x_k, u_k) + m(x_N) \quad (13)$$

$$\text{s.t.} \quad \{ x_{k+1} = f(x_k, u_k) \}.$$

such that problem (10) can be written in the form  $\min_u \{L(u, y) + E(u, y, \hat{\Sigma})\}$ . The fact that  $E$  is Lipschitz-differentiable with respect to  $u$  follows from the fact that  $f$  is assumed to be three-times Lipschitz-continuously differentiable and the assumption that  $\gamma$  is sufficiently small, which implies that the function  $Y$  is invertible in the neighborhood of a regular minimizer as established in [21]. Moreover,  $L$  is three times continuously differentiable in  $u$ . Consequently, the first order necessary optimality condition

$$0 = \nabla_u L(u^*, y) + \nabla_u E(u^*, y, \hat{\Sigma}) \implies \sigma = -\nabla_u L(u^*, y).$$

must be satisfied, which implies that  $u^*$  is a stationary point of Problem (11). Moreover, as  $l_\tau$  is a strictly convex function and  $f(0, 0) = 0$ , the Hessian matrix  $\nabla_u^2 L(0, 0)$  is positive definite, which in turn implies that

$$\nabla_u^2 L(u^*, y) = \nabla_u^2 L(0, 0) + \mathbf{O}(\gamma)$$

is positive definite, too, for sufficiently small  $\gamma$ . This follows from the fact that  $L$  is three times continuously differentiable. Thus,  $u^*$  is a regular local minimizer of the optimization problem

$$\min_u \left\{ L(u, y) + \sigma^\top u \right\},$$

which is the statement of the lemma.  $\square$

Notice that Lemma 1 is only a technical intermediate result, which is in this form not useful yet, as it relies on the availability of the (in a real-time context not available) optimal solution  $u^*$  of the optimization problem (10) in order to compute the optimal perturbation vector  $\sigma$ . However, the main idea of the current paper is to develop a real-time algorithm that computes gradient updates of the form  $\sigma = \nabla_u E(u, y, \hat{\Sigma})$  at a suitable real-time control iterate  $u$  rather than computing the optimal perturbation vector  $\nabla_u E(u^*, y, \hat{\Sigma})$ . Before we analyze this idea in more detail, the following section first focusses on an algorithm for evaluating this gradient efficiently and in real-time mode.

#### 4.2. Fast computation of the gradient

In order to develop an efficient algorithm for evaluating the gradient of the function  $E$ , it is convenient to introduce the stacked notation

$$\begin{aligned} \kappa_k &= \begin{pmatrix} x_k \\ \text{vec}(\Sigma_k) \end{pmatrix}, & \mathcal{F}(\kappa_k, u_k) &= \begin{pmatrix} f(x_k, u_k) \\ \text{vec}(F(x_k, u_k, \Sigma_k)) \end{pmatrix} \\ \omega_k &= \text{vec}(\Omega_k), & \mathcal{G}(\kappa_k, \omega_{k+1}, u_k) &= \text{vec}(G(x_k, u_k, \Omega_{k+1})) \\ \mathcal{L}(\kappa_k, \omega_{k+1}, u_k) &= \frac{1}{2} \text{Tr}(\Phi(x_k, u_k, \Omega_{k+1}) \Sigma_k), \end{aligned} \quad (14)$$

where the function “vec” stacks all independent components of a matrix into a vector exploiting symmetry whenever possible. By using this notation the function  $E$  can be written in the more compact form

$$\begin{aligned} E(u, \kappa_0) &= \min_{\kappa, \omega} \sum_{k=0}^{N-1} \mathcal{L}(\kappa_k, \omega_{k+1}, u_k) \\ \text{s.t.} \quad &\begin{cases} \kappa_{k+1} = \mathcal{F}(\kappa_k, u_k), \\ \omega_k = \mathcal{G}(\kappa_k, \omega_{k+1}, u_k), \quad \omega_N = \mathcal{M}(\kappa_N). \end{cases} \end{aligned} \quad (15)$$

A four-sweep algorithm for computing the gradient of  $E$  for a given input vector  $u$  is outlined in Algorithm 1. Notice that Algorithm 1 proposes a mixed forward-

---

**Algorithm 1: Four-sweep algorithm for updating the perturbation vector  $\sigma$ .**


---

**Input:** Control vector  $u$ ; initial state  $x_0$  and initial variance  $\Sigma_0$ .

**Four-sweeps (with  $k \in \{0, \dots, N-1\}$ ):**

1. *Nominal forward sweep.* Set  $\kappa_0 = [x_0, \text{vec}(\Sigma_0)]$  and iterate forwards

$$\kappa_{k+1} = \mathcal{F}(\kappa_k, u_k) .$$

2. *Nominal backward sweep.* Set  $\omega_N = \text{vec}(\Omega_N)$  and iterate backwards

$$\omega_k = \mathcal{G}(\kappa_k, \omega_{k+1}, u_k) .$$

3. *Adjoint forward sweep.* Set  $a_0 = 0$  and iterate forwards

$$a_{k+1} = \nabla_{\omega} \mathcal{G}(\kappa_k, \omega_{k+1}, u_k) a_k - \nabla_{\omega} \mathcal{L}(\kappa_k, \omega_{k+1}, u_k) .$$

4. *Adjoint backward sweep.* Set  $b_N = \nabla \mathcal{M}(\kappa_N) a_N$  and iterate backwards

$$b_k = \nabla_{\kappa} \mathcal{G}(\kappa_k, \omega_{k+1}, u_k) a_k + \nabla_{\kappa} \mathcal{F}(\kappa_k, u_k) b_{k+1} - \nabla_{\kappa} \mathcal{L}(\kappa_k, \omega_{k+1}, u_k) .$$

5. *Final evaluation of the gradient.* Set

$$\sigma_k = \nabla_u \mathcal{L}(\kappa_k, \omega_{k+1}, u_k) - \nabla_u \mathcal{G}(\kappa_k, \omega_{k+1}, u_k) a_k - \nabla_u \mathcal{F}(\kappa_k, u_k) b_{k+1} .$$

**Output:** The perturbation vector  $\sigma = [\sigma_0^\top, \dots, \sigma_{N-1}^\top]^\top$ .

---

backward algorithmic differentiation scheme that exploits the particular structure of the function  $E$ . The derivation of this algorithm exploits the concept of duality in linear programming as summarized in the following theorem.

**Theorem 1.** *If  $f$  is three times continuously differentiable, then Algorithm 1 returns the gradient vector  $\sigma = \nabla_u E(u, y, \Sigma_0)$ .*

*Proof.* The directional forward derivative  $\nabla_u E(u, y, \Sigma_0)^\top \Delta u$  of the function  $E$  in direction  $\Delta u$  can be found by solving the following first order linear variation of

problem (15):

$$\begin{aligned}
& \nabla_u E(u, y, \Sigma_0)^\top \Delta u = \\
& \min_{\Delta \kappa, \Delta \omega} \quad \sum_{k=0}^{N-1} \begin{pmatrix} \nabla_\kappa \mathcal{L}(\kappa_k, \omega_{k+1}, u_k) \\ \nabla_\omega \mathcal{L}(\kappa_k, \omega_{k+1}, u_k) \\ \nabla_u \mathcal{L}(\kappa_k, \omega_{k+1}, u_k) \end{pmatrix}^\top \begin{pmatrix} \Delta \kappa_k \\ \Delta \omega_{k+1} \\ \Delta u_k \end{pmatrix} \\
& \text{s.t.} \quad \begin{cases} \Delta \kappa_0 = 0, \\ \Delta \kappa_{k+1} = \nabla_\kappa \mathcal{F}(\kappa_k, u_k)^\top \Delta \kappa_k + \nabla_u \mathcal{F}(\kappa_k, u_k)^\top \Delta u_k, \\ \Delta \omega_N = \nabla \mathcal{M}(\kappa_N)^\top \Delta \kappa_N, \\ \Delta \omega_k = \begin{pmatrix} \nabla_\kappa \mathcal{G}(\kappa_k, \omega_{k+1}, u_k) \\ \nabla_\omega \mathcal{G}(\kappa_k, \omega_{k+1}, u_k) \\ \nabla_u \mathcal{G}(\kappa_k, \omega_{k+1}, u_k) \end{pmatrix}^\top \begin{pmatrix} \Delta \kappa_k \\ \Delta \omega_{k+1} \\ \Delta u_k \end{pmatrix}. \end{cases} \quad (16)
\end{aligned}$$

By writing out the first order optimality conditions of the above linear programming problem, explicit expressions for the multipliers  $a_0, a_1, \dots, a_N$  as well as  $b_N, b_{N-1}, \dots, b_0$  are found. They are given by the recursion in Step 3 and 4 of Algorithm 1. Moreover, since Problem (16) is a linear programming problem, there is no duality gap, i.e., the objective values of the primal and dual objectives coincide,

$$\begin{aligned}
& \nabla_u E(u, y, \Sigma_0)^\top \Delta u = \\
& \sum_{k=1}^{N-1} \Delta u_k^\top [\nabla_u \mathcal{L}(\kappa_k, \omega_{k+1}, u_k) - \nabla_u \mathcal{G}(\kappa_k, \omega_{k+1}, u_k) a_k - \nabla_u \mathcal{F}(\kappa_k, u_k) b_{k+1}].
\end{aligned}$$

As this equation holds for all directions  $\Delta u$ , a comparison of coefficients yields  $\sigma = \nabla_u E(u, y, \Sigma_0)$ , as stated by the theorem.  $\square$

#### 4.3. Real-time algorithm

A real-time iteration scheme for self-reflective MPC is obtained by implementing the following steps in a loop.

1. Compute the gradient  $\sigma = \nabla_u E(u, x_0, \hat{\Sigma})$  by using Algorithm 1.
2. Collect new measurements and use an extended Kalman filter to update  $y$ .
3. Compute a local minimizer  $(x^+, u^+)$  of the optimization problem (11).
4. Send the solution  $u_0^+$  to the process.
5. Shift all variables,  $u_k \leftarrow u_{k+1}^+$ ,  $u_{N-1} = u_{N-1}^+$ ,  $x_k \leftarrow x_{k+1}^+$ , and  $x_N = x_N^+$ .
6. Update the variance matrix  $\hat{\Sigma} \leftarrow F(x_0^+, u_0^+, \hat{\Sigma})$ .



Notice that in the above scheme, the perturbation vector  $\sigma$  is updated in the preparation phase in Step 1, i.e., before the actual measurement arrives. This is motivated by the fact that the computation of this gradient is usually the most expensive step and therefore done before the feedback phase, Step 2-4. Notice that variants of the above scheme might refine Step 3 and solve a quadratic approximation of (11) in order to further speed-up the feedback time, which leads to a scheme that is similar to the standard real-time algorithms in Sections 2.2 and 2.3. However, before we discuss such variants, let us first analyze, why the above real-time scheme can be expected to be contractive at all. For this aim, we introduce the auxiliary function<sup>2</sup>

$$\phi(u) = \underset{v}{\operatorname{argmin}} \left\{ L(v, y) + \nabla_u E(u, y, \hat{\Sigma})^\top v \right\}. \quad (17)$$

Notice that Lemma 1 implies that the solution  $u^*$  of the self-reflective optimization problem (10) is a fix-point of the function  $\phi$ ,  $u^* = \phi(u^*)$ . Moreover, the update from Step 3 of the above outlined real-time procedure can be written in the form

$$u^+ = \phi(u).$$

Thus, the proposed algorithm can be interpreted as a real-time variant of a fixed point iteration. Now, a contractivity condition can be established by using Banach's fixed point theorem.

**Theorem 2.** *Let all the conditions from Lemma 1 be satisfied and let the current iterate  $u$  for the control input be in a neighborhood of  $u^*$ . The update from Step 3 of the proposed real-time scheme contracts linearly, i.e., we have*

$$\|u^+ - u^*\| \leq c \|u - u^*\|$$

with contraction constant  $c < 1$ .

*Proof.* Lemma 1 assumes that the local minimizer  $u^*$  is regular. Consequently, the optimization problem (17) is regular for all  $u$  in a sufficiently small neighborhood

---

<sup>2</sup>Notice that (if  $L$  is not strictly convex in  $v$ ) (17) introduces a slight abuse of notation in the sense that this paper uses local optimization algorithms only. In the context of this paper, (17) should be read as “ $\phi(u)$  is the local minimizer that is found by initializing the local solver in a neighborhood of the solution  $u^*$  of (10)”.

of  $u^*$ . By using a standard result from parametric nonlinear programming [32], this implies that the map  $\phi$  is locally Lipschitz continuous with respect to the perturbation gradient,

$$\phi(u_1) - \phi(u_2) = \mathbf{O}(\|\nabla_u E(u_1, y, \hat{\Sigma}) - \nabla_u E(u_2, y, \hat{\Sigma})\|)$$

for all  $u_1, u_2$  in a neighborhood of  $u^*$ . Next, we can use that the gradient  $\nabla_u E$  is Lipschitz continuous, which implies

$$\phi(u_1) - \phi(u_2) = \mathbf{O}(\gamma) \|u_1 - u_2\| ,$$

since  $E$ —and consequently also the Lipschitz constant of  $\nabla_u E$ —is of order  $\mathbf{O}(\gamma)$ . Thus, we have

$$\|u^+ - u^*\| = \|\phi(u) - \phi(u^*)\| \leq c \|u - u^*\| ,$$

with  $c = \mathbf{O}(\gamma)$ , i.e., the update contracts if  $\gamma$  is sufficiently small.  $\square$

As any other gradient based optimization method, also the performance of the above outlined real-time scheme can be improved further by implementing a preconditioner [32]. In this paper, the optimization problem (10) is solved once offline at the steady-state. The corresponding Hessian matrix,  $\nabla_u^2 E(0, 0, \gamma^2 I)$ , or a suitable approximation of this matrix, can be used to scale the optimization variable  $u$  offline in order to improve the contraction rate of the proposed real-time scheme. As mentioned above, another improvement of the above real-time scheme refines Step 3 by solving a quadratic approximation of (11) at every sampling time instead of solving this problem to optimality. In this case, the evaluation of all gradients can be done in the preparation step, as explained in Section 2.2. A discussion of why contractivity of the real-time iterates is enough to ensure nominal local closed-loop stability of the corresponding MPC controller for exact state measurements and under mild regularity assumptions can be found in [6]. However, in the context of this paper, we need to make the additional assumption that the system is locally observable such that the extended Kalman filter yields bounded variance matrices of order  $\hat{\Sigma} = \mathbf{O}(\gamma^2)$ ; see also [7, 13] for an overview of how to analyze the closed loop-stability of MPC-EKF cascades.

## 5. Numerical case study

### 5.1. Chemical process model

This paper analyzes a controllable chemical reaction, where the discrete-time right-hand side function  $f(x, u) = z(h, x, u)$  is given in the form of the solution

of the differential equation system

$$\frac{\partial z(t, x, u)}{\partial t} = \begin{pmatrix} -(D + k_1)z_1(t, x, u) - k_2z_2(t, x, u)z_3(t, x, u) + u_1 \\ -Dz_2(t, x, u) - k_3z_2(t, x, u)z_3(t, x, u) + k_4z_1(t, x, u) + u_2 \\ -Dz_3(t, x, u) - k_5z_2(t, x, u)z_3(t, x, u) + u_3 \end{pmatrix}$$

$$z(0, x, u) = x,$$

which has to be evaluated numerically, e.g., by using a Runge-Kutta integrator. The given constant  $h = \frac{1}{2}$  denotes the discrete-time step-size. The differential states  $z_1, z_2$  and  $z_3$  denote the concentrations of three substances, which react with each other. For simplicity of presentation, the corresponding reaction constants  $k_1 = k_2 = k_4 = \frac{1}{2}$  and  $k_3 = k_5 = \frac{1}{10}$  as well as the dilution rate  $D = 0.1$  are assumed to be given. The feeding rates  $u_1, u_2$ , and  $u_3$  are control inputs, which can be used to adjust the inflow of the substances. We assume that only the concentration  $z_1$  can be measured,  $C = [1, 0, 0]^\top$ . The variance of the measurement error is given by  $V = 0.005^2$ . A summary of all model parameters can be found in Table 1.

Name	Symbol	Value
discrete-time step-size	$h$	0.5
MPC horizon length	$N$	20
lower bound on the control	$\underline{u}$	$[0, -1, 0]^\top$
upper bound on the control	$\bar{u}$	$[\infty, \infty, \infty]$ (not implemented)
initial state estimate	$\hat{y}$	$[1, 5, 0]^\top$
state reference	$x_{\text{ref}}$	$[1, 5, 0]^\top$
control reference	$u_{\text{ref}}$	$[0.6, 0, 0]^\top$
initial state variance	$\hat{\Sigma}$	0
measurement error variance	$V$	$2.5 \cdot 10^{-5}$
process noise variance	$W$	$\text{diag}(0, 0.64, 0)$
state weighting matrix	$Q$	$\text{diag}(1, 1, 1)$
control weighting matrix	$R$	$\text{diag}(1, 1, 100)$
terminal cost weighting matrix	$P_N$	$\text{diag}(1, 1, 1)$

Table 1: Parameter values.

The stage and terminal cost functions are given by the quadratic expressions

$$l(x, u) = \frac{1}{2} \left( (x - x_{\text{ref}})^\top Q (x - x_{\text{ref}}) + (u - u_{\text{ref}})^\top R (u - u_{\text{ref}}) \right)$$

and

$$m(x) = \frac{1}{2} (x - x_{\text{ref}})^\top P_N (x - x_{\text{ref}}).$$

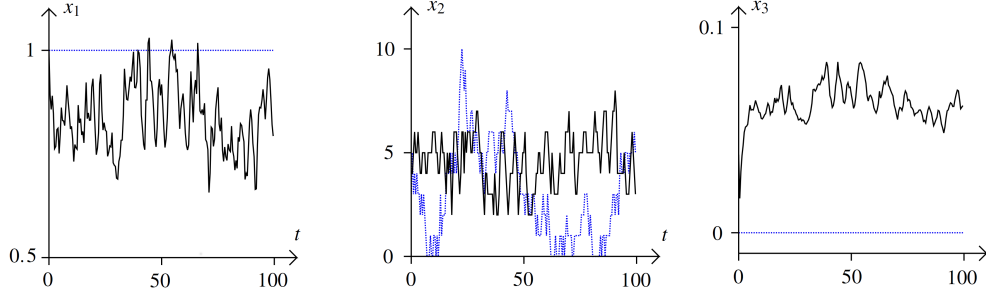


Figure 1: Closed-loop control trajectories of certainty-equivalent MPC (blue, dotted) and self-reflective MPC (black, solid) in the presence of random process noise and measurement errors.

At this point, we have introduced a small abuse of notation as in the previous sections we have set  $x_{\text{ref}} = 0$  and  $u_{\text{ref}} = 0$ . However, the above problem can easily be brought into this form by shifting all states and controls by a constant offset.

### 5.2. Implementation details

The implementation in this paper is based on the real-time algorithm from Section 4.3. Here, a pre-conditioner is implemented by computing the exact Hessian of  $E$  offline at the steady-state, as explained in Section 4.3, and we implement the refined version of Step 3, i.e., only a quadratic approximation of Problem (11) is solved at every sampling time. The barrier parameter  $\tau$  is kept constant and set to  $\tau = 0.001$ , which turns out to lead to sufficiently accurate results for this particular case study. Moreover, we use automatic code generation tools in order to export all algorithmic routines in the form of optimized C-code [19]. In particular, the four-sweep method for the perturbation vector update, Algorithm 1, is implemented by using the algorithmic differentiation software CASADI [1, 2]. Additionally, we use the software qpOASES [9] as a QP solver for implementing the real-time update. All the results below are obtained on a Mac OS X EI Captain operating system with 2.6 GHz processor and 8 GB, 1600 MHz DDR3.

### 5.3. Control performance

Figure 1 shows a comparison of the closed loop state trajectories for randomly chosen uncertainty scenarios. The blue dotted lines in the left, middle, and right plot show the closed loop trajectories that have been obtained by running a certainty equivalent MPC controller in combination with an extended Kalman filter. We have simulated uniformly distributed process noise and measurement errors.

The corresponding variance matrices  $W$  and  $V$  are listed in Table 1. Notice that the certainty equivalent MPC controller keeps the states  $x_1$  and  $x_3$  very close to their steady-state values, as this controller does not detect a need to excite the system. This is contrast to the self-reflective MPC controller, which excites the states  $x_3$  and  $x_1$  in order to be able to estimate all states more accurately. Recall that only the first state  $x_1$  can be measured in our example and thus an excitation is necessary in order to be able to gather information about the other states. The main difference between the certainty equivalent MPC controller and the self-reflective MPC controller can be seen when looking at the state  $x_2$ . While the self-reflective MPC controller is able to keep this state in a neighborhood of its reference state  $[x_{\text{ref}}]_2 = 5$ , the corresponding closed-loop trajectory of the certainty equivalent controller leads to much larger deviations in the presence of noise, as almost no information about this state is available when keeping  $x_1$  and  $x_3$  close to their steady-state values. This difference between the two controllers can be illustrated by analyzing their average performance. For the certainty-equivalent MPC controller the value

$$\frac{1}{M} \sum_{i=1}^M l(x_i^{\text{MPC}}, u_i^{\text{MPC}}) \approx 11.4$$

is found for sufficiently long closed-loop simulation times  $M$ . The corresponding average performance of the self-reflective MPC controller is given by

$$\frac{1}{M} \sum_{i=1}^M l(x_i^{\text{SRMPC}}, u_i^{\text{SRMPC}}) \approx 2.2$$

using the same random uncertainty scenario. Of course, the main reason why the performance difference between the two controllers is for this example particularly large is mainly due to the fact that the considered control system is not observable at its steady-state such that the self-reflective excitations lead to a major performance improvement in the presense of noise. For other applications the improvement of control performance may be less spectacular, but in general self-reflective MPC performs better than certainty-equivalent MPC and therefore, at least for some applications, this type of self-reflective controllers should be of practical interest.

#### 5.4. Runtime performance

Table 2 summarizes the run-time of the different steps of the proposed real-time algorithm for self-reflective MPC. The computation of the perturbation vector up-

	CPU time ( $\mu\text{s}$ )	%
Computation of the perturbation vector update (Algorithm 1)	325	68
Preparation of the QP (evaluation of $A$ , $B$ , etc.)	41	9
Computation of $x^+$ and $u^+$ with qpOASES (feedback step)	108	22
Other operations (including Kalman filter update)	$\leq 3$	$\approx 1$
Total time	477	100
Comparison: total CPU time of real-time MPC without learning term	152	32

Table 2: CPU time of one real-time step.

date takes approximately  $325\mu\text{s}$ , which corresponds to about 68% of the overall CPU time needed for one real-time step of the proposed self-reflective MPC algorithm. A standard MPC controller without learning terms needs  $152\mu\text{s}$  per real-time step, as such a standard controller needs to implement basically the same operations except for the perturbation vector update. Given the fact that solving the original self-reflective optimization problem (10) with a generic optimal control solver takes more than 800 times longer than the standard MPC problem, the above run-times must be considered as a major improvement compared to the state-of-the-art, although one might still argue that an increase of a factor 3-4 in terms of run-time for adding a self-reflective learning term correction is still a lot.

**Remark 1.** *The case study in this paper has only three states and must be considered a small-scale example. Thus, there arises the question how the proposed real-time algorithm performs for larger scale examples. Although we do not present such larger case studies in detail as part of this paper, preliminary numerical experiments indicate that also for larger problems one real-time step of the proposed algorithm takes approximately three to four times longer than a standard real-time MPC step, i.e., also for larger problems a relatively moderate increase in run-time is observed if the additive self-reflective learning term is taken into account. Such a behavior can also be expected from our theoretical considerations in the sense that both the standard MPC real-time step applied to a problem formulation without learning term as well as Algorithm 1 have a complexity of order  $\mathcal{O}(Nn_x^3)$  recalling that  $n_x$  denotes the state dimension and  $N$  the horizon length.  $\diamond$*

## 6. Conclusion

This paper has proposed a novel real-time algorithm for integrated experiment design MPC. This algorithm improves the run-time performance compared to ex-

isting generic real-time MPC algorithms by orders of magnitude, as it is capable of exploiting the particular structure of MPC problems with additional learning terms. A particular focus of this paper has been on self-reflective MPC, a controller whose objective is to minimize the sum of a nominal tracking term and the expected loss of optimality of its own performance in the presence of random process noise and measurement errors. Lemma 1 has established that such self-reflective model predictive controllers are equivalent to a standard MPC problem with affinely perturbed objective function as long as the perturbation vector is chosen to be equal to the gradient of the expected loss of optimality. As this affinely perturbed MPC problem can be solved as fast as standard MPC problems without learning terms, the only additional cost is the cost of computing or approximating the optimal perturbation vector. A four-sweep algorithm for computing this perturbation vector approximately and in real-time has been proposed in Algorithm 1. The properties of the associated overall real-time scheme for self-reflective MPC have been analyzed in Theorem 2 and have been illustrated numerically by applying it to a nonlinear predator-prey-feeding control problem. For the presented case study, one real-time step takes  $325\text{ }\mu\text{s}$ , approximately three times more than a standard real-time MPC step without learning perturbation. However, compared to existing generic real-time MPC methods applied to the self-reflective problem this amounts to a moderate increase in run-time that may be acceptable in practice, if the additional learning terms leads—as in the presented example—to a significantly improved overall control performance in the presence of measurement errors and process noise.

## Acknowledgements

This research was supported by National Natural Science Foundation China, Nr. 61473185, as well as ShanghaiTech University, Grant-Nr. F-0203-14-012.

## References

## References

- [1] J. Andersson, B. Houska, M. Diehl. Towards a Computer Algebra System with Automatic Differentiation for Use with Object-Oriented Modelling Languages. *3rd International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, pp. 99–105, Oslo, Norway, October 3, 2010.

- [2] J. Andersson, J. Akesson, M. Diehl. CasADi: A Symbolic Package for Automatic Differentiation and Optimal Control. In *Recent Advances in Algorithmic Differentiation*, Volume 87 of the series Lecture Notes in Computational Science and Engineering, p. 297–307, 2012.
- [3] L.T. Biegler. An overview of simultaneous strategies for dynamic optimization. *Chemical Engineering and Processing*, 46:1043–1053, 2007.
- [4] H.G. Bock, M. Diehl, D.B. Leineweber, and J.P. Schlöder. A direct multiple shooting method for real-time optimization of nonlinear DAE processes. In F. Allgöwer and A. Zheng, editors, *Nonlinear Predictive Control*, volume 26 of *Progress in Systems Theory*, pages 246–267, Basel Boston Berlin, 2000. Birkhäuser.
- [5] S. Boyd and L. Vandenberghe. *Convex Optimization*. University Press, Cambridge, 2004.
- [6] M. Diehl, H.G. Bock, J.P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and Nonlinear Model Predictive Control of Processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585, 2002.
- [7] M. Diehl and H.J. Ferreau and N. Haverbeke. *Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation*, In *Nonlinear Model Predictive Control*, 384:391–417, 2009.
- [8] A.A. Feldbaum. Dual-control theory (I-IV). *Automation and Remote Control*, 21, pages 1240–1249 and 1453–1464, 1960, as well 22, pages 3–16 and 129–143, 1961.
- [9] H. J. Ferreau, H. G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, 18(8):816–830, 2008.
- [10] N.M. Filatov, H. Unbehauen. *Adaptive Dual Control*. Springer, 2004.
- [11] R. Findeisen, L. Imsland, F. Allgöwer, and B.A. Foss. *State and Output Feedback Nonlinear Model Predictive Control: An Overview*, *European Journal of Control*, 9:190–207, 2003.



- [12] A. Griewank. *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation*. Number 19 in Frontiers in Appl. Math. SIAM, Philadelphia, 2000.
- [13] N. Haverbeke. Efficient Numerical Methods for Moving Horizon Estimation. Ph.D. Thesis, KU Leuven, 2011. ISBN 978-94-6018-417-8.
- [14] T.A.N. Heirung, B.E. Ydstie, B. Foss. Towards Dual MPC. In Proceedings of the 4th IFAC Nonlinear Model Predictive Control Conference, pp:502–507, Noordwijkerhout (Netherlands), 2012.
- [15] T.A.N. Heirung, B. Foss, B.E. Ydstie. MPC-based dual control with online experiment design. *Journal of Process Control*, 32, 64–76, 2015.
- [16] B. Hernandez, P. Trodden. Persistently exciting tube MPC. arXiv:1505.05772v1, 2015.
- [17] H. Hjalmarsson. System identification of complex and structured systems. *European Journal of Control*, 15, 275–310, 2009.
- [18] M. Hovd, R.R. Bitmead. Interaction between control and state estimation in nonlinear MPC. *Modeling, Identification, and Control*, 26(3):165–174, 2005.
- [19] B. Houska, H.J. Ferreau, and M. Diehl. An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. *Automatica*, 47(10):2279-2285, 2011.
- [20] B. Houska, D. Telen, F. Logist, M. Diehl and J. Van Impe. An Economic Objective for Optimal Experiment Design of Nonlinear Dynamic Processes. *Automatica*, 51, pp:98–103, 2015.
- [21] B. Houska, D. Telen, F. Logist, J. Van Impe. Self-reflective model predictive control. arXiv:1610.03228, 2016.
- [22] C.A. Larsson, M. Annergren, H. Hjalmarsson, C.R. Rojas, X. Bombois, A. Mesbah, P. Moden. Model predictive control with integrated experiment design for output error systems. In proceedings of the 2013 European Control Conference (ECC), 3790–3795, July, 2013.

- [23] C. Larsson, C. Rojas, X. Bombois, H. Hjalmarsson. Experimental evaluation of model predictive control with excitation (MPC-X) on an industrial depropanizer. *Journal of Process Control*, 31:1–16, 2015.
- [24] J.M. Lee and J.H. Lee. An approximate dynamic programming based approach to dual adaptive control. *Journal of Process Control*, 19(5):859–864, 2009.
- [25] G. Marafioti, R.R. Bitmead, M. Hovd. Persistently exciting model predictive control. *International Journal of Adaptive Control and Signal Processing*, 28(6):536–552, 2014.
- [26] A. Marchetti, B. Chachuat, D. Bonvin. Modifier-adaptation methodology for real-time optimization. *Industrial & Engineering Chemistry Research*, 48(13):6022–6033, 2009.
- [27] J. Mattingley and S. Boyd. Automatic Code Generation for Real-Time Convex Optimization. *Convex Optimization in Signal Processing and Communications*, Y. Eldar and D. Palomar, Eds., Cambridge University Press, 2009.
- [28] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.Q. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6), 789–814, 2000.
- [29] D.Q. Mayne, S.V. Rakovic, R. Findeisen, and F. Allgöwer. Robust output feedback model predictive control of constrained linear systems. *Automatica*, 42:1217–1222, 2006.
- [30] D.Q. Mayne, S.V. Rakovic, R. Findeisen, and F. Allgöwer. Robust output feedback model predictive control of constrained linear systems: Time varying case. *Automatica*, 45:2082–2087, 2009.
- [31] A. Mesbah, X. Bombois, M. Forgone, H. Hjalmarsson, P.M.J. Van den Hof. Least costly losed-loop performance diagnosis and plant re-identification. *International Journal of Control*, 88(11):22642276, 2015.
- [32] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2 edition, 2006.
- [33] L.S. Pontryagin, V.G. Boltyanskii, R.V. Gamkrelidze, E.F. Mishchenko. *The Mathematical Theory of Optimal Processes*. John Wiley & Sons, New York, 1962.

- [34] F. Pukelsheim. Optimal design of Experiments. John Wiley & Sons, Inc., New York, 1993.
- [35] S.J. Qin and T.A. Badgwell. A survey of industrial model predictive control technology. *Control engineering practice*, 11(7), 733–764, 2003.
- [36] R. Quirynen, B. Houska, M. Vallerio, D. Telen, F. Logist, J. Van Impe, M. Diehl. Symmetric algorithmic differentiation based exact Hessian SQP method and software for economic MPC. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 2752–2757, 2014.
- [37] J.B. Rawlings and D.Q. Mayne. *Model Predictive Control: Theory and Design*. Madison, WI: Nob Hill Publishing, 2009.
- [38] S. Sager. Sampling decisions in optimum experimental design in the light of Pontryagin’s maximum principle. *SIAM Journal on Control and Optimization* 51(4), 3181–3207, 2013.
- [39] M.S. Shouche, H. Genceli, M. Nikolaou. Effect of online optimization techniques on model predictive control and identification (MPCI). *Computers & Chemical Engineering*, 26(9):1241–1252, 2002.
- [40] R. Stengel. *Optimal Control and Estimation*. Dover Publications, New York, 1994.
- [41] D. Telen, B. Houska, F. Logist, E. Van Derlinden, M. Diehl, J. Van Impe. Optimal experiment design under process noise using Riccati differential equations. *Journal of Process Control*, 23:613–629, 2013.
- [42] B. Wittenmark. Adaptive dual control methods: An overview. In *5th IFAC symposium on Adaptive Systems in Control and Signal Processing Budapest, Hungary*, 1995.
- [43] S.J. Wright. *Primal-dual interior-point methods*. SIAM, 1997.
- [44] E. Zacekova, S. Privara, M. Pcolka. Persistent excitation condition within the dual control framework. *Journal of Process Control*, 23(9):1270–1280, 2013.
- [45] V. M. Zavala and L.T. Biegler. The Advanced Step NMPC Controller: Optimality, Stability and Robustness. *Automatica*, 45:86–93, 2009.